Embedding of Tree Machines into Hypercubes

Chui-Cheng Chen
Department of Information Management
Southern Taiwan University of Technology
ccchen@mail.stut.edu.tw

Abstract

In this paper, we present that a tree machine can be embedded into incomplete hypercube with expansion 1, load 1, dilation 2 and congestion 2. This result is better than the expansion $(2^{h+1}+2^h)/(2^{h+1}+2^h-2)$ *in [14]. Then we consider how to embed a large tree machine into a hypercube for considering load-balance. We have shown that a tree machine* TM_h $(h\geq 1)$ *can be embedded into a hypercube* H_{h+1} *with dilation 1, congestion 2 and load 2, and a tree machine* TM_h *can be embedded into* H_n $(h\geq n\geq 1)$ *with dilaton 1, congestion 3 and load* $2^{h-n+1}+2^{h-n}$. *The load of these embeddings is well balanced.*

1 Introduction

The hypercubes [1, 2] is one of the most popular architectures for a variety of parallel computations. The architectural features are its low diameter, rich bandwidth, regular structure, easy routing, fault tolerance [3, 4] and, more importantly, many computational structures can be simulated by the hypercubes with only small constant factor slowdown, such as array, binary tree and mesh of trees [5].

One restriction of the hypercube topology is that its size has to be an integer power of two. Hence, two consecutive dimensional hypercubes leave a large gap. To overcome this restriction, incomplete hypercubes provide more flexibility in the size [6]. An incomplete hypercube can be obtained from a complete hypercube where some nodes/links fail. Extensive study on the incomplete hypercube has been studied in [7-9].

The tree machine can permit efficient algorithms for searching problems [10, 11] and for graph problems [12]. Efe [13] has shown that a tree machine of dimension h is a subgraph in an (h+2)-dimensional hypercube. Öhring and Das [14] have described how to embed a tree machine of dimension h into an incomplete hypercube which comprises an (h+1)-dimensional hypercube and an h-dimensional hypercube with *load* 1, *dilation* 2 and *congestion* 2.

The purpose of this paper is to present how to embed a tree machine into an incomplete hypercube and a hypercube. First, we discuss how to embed a tree machine into an incomplete hypercube with *load* 1, *dilation* 2, *congestion* 2 and *expansion* 1, then discuss how to embed a large tree machine into a hypercube with optimal load.

The remaining sections are organized as follows. Section 2 gives the notations and definitions of this paper. In Section 3, we present how to embed a tree machine into an incomplete hypercube with the same node size (*expansion* 1). In Section 4, we present how to embed a large tree machine into a hypercube. Finally, the conclusion is given in Section 5.

2 Preliminaries

We let H_n denote the n-dimensional hypercube which has 2^n nodes. These nodes of H_n are labeled {0, 1, 2, …, 2^n-1} as binary number. Let $(1*^{n-1})$ denote a sub-hypercube with the most significant bit labeled 1, where $*$ is a don't care symbol assuming 0 or 1. There is an edge between two nodes in the hypercube if and only if their binary numbers differ by a single bit. The *Hamming distance* between two nodes is defined as the number of their different bits. To conveniently describe the embedding, we use two colors, black and white, to correspond to the binary number of each node in H_n. If a node has even number of 1's, we color it black. Otherwise we color the node white. Since the hypercube has a perfect matching, H_n has 2^{n-1} black nodes and 2^{n-1} white nodes. Figure 1 depicts the four-dimensional hypercube, H_4. However, the size of the hypercube has to be a power of 2, whereas it is more interesting to design an architecture with an arbitrary number of nodes.

A complete hypercube can be come incomplete if it misses some certain nodes [6-9]. Let $IH(n1, n2, \ldots, ni)$ denote the incomplete hypercube comprising i complete hypercubes: H_{n1}, H_{n2}, …, H_{nj}, …, H_{ni}, $nj>ni\geq 0$, which can be obtained by deleting the largest $2^{n1}-(2^{n2}+\ldots+2^{ni})$ nodes (in binary representation) and their neighboring edges from an ($n1$+1)-dimensional hypercube. For example, Figure 2 depicts $IH(3, 2, 1)$.

A double-rooted complete binary tree of height *h*, denoted by DT_h, is a complete binary tree of height *h* with the root replaced by a path of length two [5]. We denote leaf nodes of left-subtree of the roots of DT_h as $\{x_1, x_2, \ldots, x_2h\text{-}1\}$ from left to right, likewise, we denote leaf nodes of right-subtree of the roots of DT_h as $\{y_1, y_2, \ldots, y_2h\text{-}1\}$ from left to right. For example, Figure 3 depicts DT_3. A tree machine of dimension *h*, TM_h, is the graph comprising two complete binary trees of height *h*, connected back to back to share the leaf nodes, The total number of TM_h is $2^{h+1}+2^h-2$ nodes. For example, Figure 4 depicts TM_3.

The cost of an embedding of a *guest graph* into a *host graph* is measured in terms of *dilation*, *congestion*, *load* and *expansion* [15]. The dilation of an edge of the *guest graph* is the length of embedded path of the *host graph*, and the *dilation* of an embedding is the maximum dilation over all edges of the *guest graph*. The *congestion* of an edge of the *host graph* is the number of edges of the *guest graph* that are embedded using the same edge of the *host graph*, and the *congestion* of an embedding is the maximum congestion over all edges of the *host graph*. The load of a node of the *host graph* is the number of nodes of the *guest graph* that are embedded into the same node of the *host graph*, and the *load* of an embedding is the maximum load over all nodes of the *host graph*. The *expansion* of an embedding is the ratio of the number of nodes of the *host graph* to the number of nodes of the *guest graph*. Hence, we need to consider the tradeoff among *dilation*, *congestion*, *load* and *expansion* of an embedding.

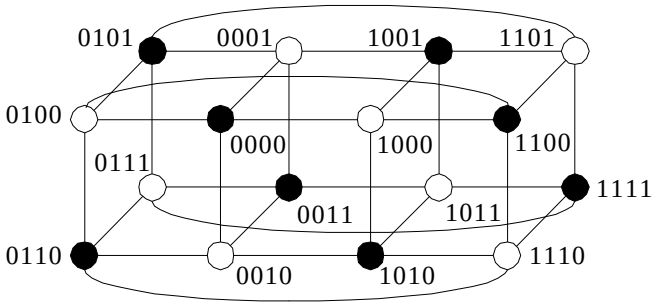


Figure 1. H_4.

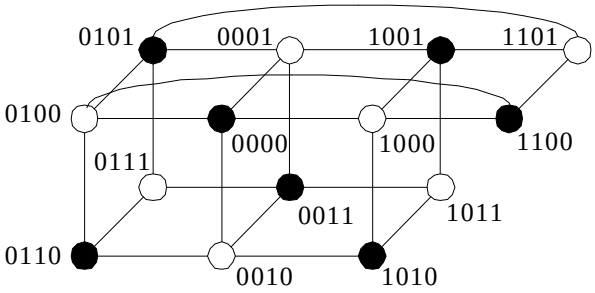


Figure 2. *IH*(3, 2, 1).

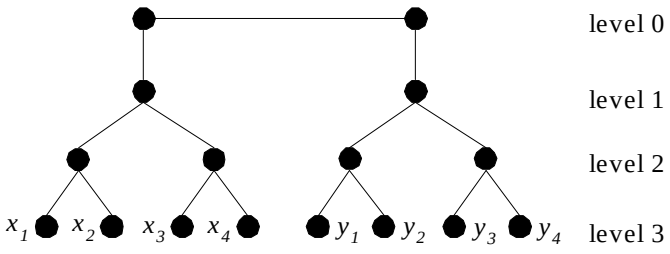


Figure 3. DT_3.

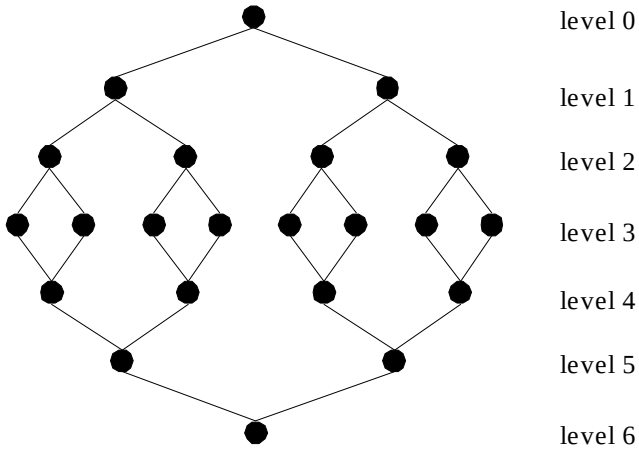


Figure 4. TM_3.

3 Embedding of Tree Machines into Incomplete Hypercubes with *Expansion* 1

Öhring and Das [14] have shown that TM_h can be embedded into $IH(h+1, h)$ with *load* 1, *dilation* 2 and *congestion* 2, and TM_h is not a subgraph of $IH(h+1, h)$. In this section, we discuss how to embed a tree machine into an incomplete hypercube with the same node size (*expansion* 1), considering the dilation and congestion when doing the embedding. For the embedding, we need the following lemma.

Lemma 1. In $IH(h+1, h)$, all nodes of H_h are placed on the outer level and all nodes of H_{h+1} are placed on the inner level. Each node of the outer level can link to two son nodes on the inner level with *dilation* 2 and *congestion* 2

Proof. We use $h+2$ bits to label each node of $IH(h+1, h)$. $IH(h+1, h)$ can be partitioned into three sub-hypercube H_h's by the leftmost two bits of the incomplete hypercube, and the binary numbers of the leftmost two bits of the three sub-hypercubes correspond to 00, 01 and 10 (see Figure 5).

We place all nodes labeled $10*^h$ on the outer (upper) level and all nodes labeled $0*^{h+1}$ on the inner (lower) level. Each node labeled $10*^h$ has an edge with dilation 1 to link to one node labeled $00*^h$ and an edge with dilation 2 to link to one node labeled $01*^h$. The congestion of edges between sub-hypercube 00 and 10 are 2. Therefore. The lemma is proved. □

One example is shown as follows.

Example 1. All the nodes of H_2 are placed on the outer level and all the nodes of H_3 are placed on the inner level in $IH(3, 2)$. Each node of the outer level can link to two son nodes on the inner level with *dilation* 2 and *congestion* 2 (see Figure 6).

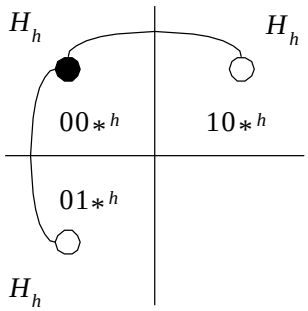


Figure 5. $IH(h+1, h)$ is partitioned into three sub-hypercube H_h's by the leftmost two bits of the incomplete hypercube.

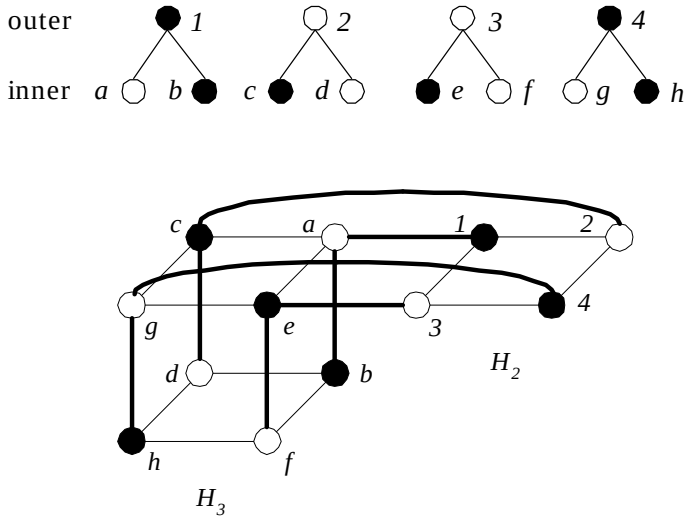


Figure 6. All nodes of H_2 and H_3 are placed on the outer level and the inner level, respectively. The dilation of edges (1, *b*), (2, *d*), (3, *f*) and (4, *h*) are 2 between the outer level and the inner level. The congestion of edges (1, *a*), (2, *c*), (3, *e*) and (4, *g*) are 2 in *IH*(3, 2).

Theorem 1. The tree machine TM_h (h≥1) can be embedded into the incomplete hypercube *IH*(h+1, h-1, h-2, …, 3, 2, 1) with *load* 1, *dilation* 2, *congestion* 2, and *expansion* 1.

Proof. First, H_1 is partitioned into two H_0's, and the nodes on levels 0 and 2h of TM_h are embedded into the two H_0's, respectively. Next, we partitioned H_{i+1} into two sub-hypercubes H_i's by linking respectively to the two sub-hypercubes H_{i-1}'s of H_i, and the nodes on levels i and 2h-i of TM_h are embedded into the two sub-hypercubes H_i's, respectively, 1≤i≤h-1. Hence, the nodes between 0 and h-1, and between levels 2h and h+1 of TM_h can be embedded into *IH*(h, h-1, h-2, …, 3, 2, 1) with *dilation* 2 and *congestion* 2 by Lemma 1. Here, H_{h+1} is partitioned into two H_h's and one of them, called *front* H_h, is used to embed the nodes on levels h-1 and h+1 of TM_h.

We use a similar method to embed the nodes on level h of TM_h as follows. The nodes on level h of TM_h are embedded into the other H_h, called *back* H_h, The *back* H_h is partitioned into two H_{h-1}'s by linking respectively to the two sub-hypercubes H_{h-1}'s of front H_h, then the links between levels h-1 and h as well as between levels h+1 and h of TM_h are as the same as Lemma 1, but the congestion of edges between two sub-hypercubes H_{h-1}'s of *back* H_h are 2. Therefore, TM_h is embedded into *IH*(h+1, h-1, h-2, …, 3, 2, 1) with *load* 1, *dilation* 2, *congestion* 2 and *expansion* 1. □

One example is shown as follows.

Example 2. The tree machine TM_2 can be embedded into *IH*(3, 1) with *load* 1, *dilation* 2, *congestion* 2 and *expansion* 1 (see Figure 7).

4 Embedding of Large Tree Machines into Hypercubes

Efe [13] has shown that TM_{h-1} is a subgraph of H_{h+1}. In this section, we discuss how to embed a large tree machine into a hypercube, considering the load, dilation and congestion while doing the embedding. First, we embed TM_h into H_{h+1}, and a lemma is thus required as follows.

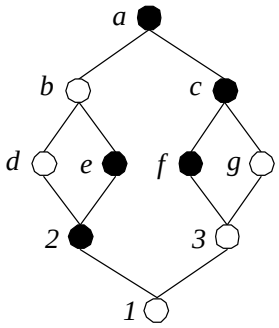
Lemma 2. Embedding DT_h (h≥1) into H_{h+1}, each leaf node x_i (1≤i≤2^{h-1}) of DT_h has an edge to link to a leaf node y_i of DT_h in H_{h+1}.

Proof. We prove the lemma by induction on h.

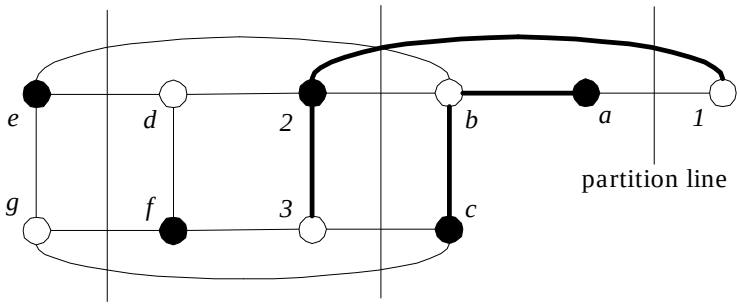
Hypothesis: Embedding DT_{h-1} into H_h, each leaf node x_i (1≤i≤2^{h-2}) of DT_{h-1} has an edge to link to a leaf node y_i of DT_{h-1} in H_h.

Basis step: When h=1 and 2, it is trivial. Figure 8 depicts the links of embedding DT_2 into H_3.

Induction step: We partition H_{h+1} into two H_h's by the most significant bit, and embed DT_{h-1} into H_h as the hypothesis above describe. We can merge two DT_{h-1}'s to DT_h as shown in Figure 9. The links between x_i (1≤i≤2^{h-1}) and y_i of DT_h are the same as the hypothesis since leaf nodes of two DT_{h-1}'s are also leaf nodes of DT_h after merging. Therefore, the lemma is proved. □



TM_2



back H_2 *front* H_2 H_1

Figure 7. Dilation of edges (*1, 3*), (*a, c*), (*2, e*), (*3, g*), (*b, d*) and (*c, f*) is 2 in TM_2. Congestion of edges (*1, 2*), (*a, b*), (*b, e*), (*c, g*), (*2, d*), (*3, f*), (*e, d*) and (*g, f*) is 2 in *IH*(3, 1).

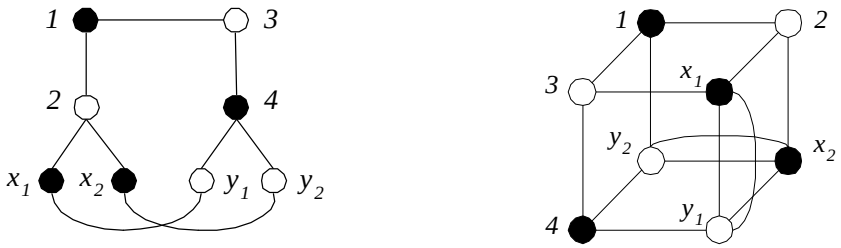


Figure 8. Leaf nodes x_1 and x_2 have edges to link respectively to y_1 and y_2 in H_3.

Now, we show how to embed TM_h into H_{h+1}.

Theorem 2. A tree machine TM_h ($h \geq 1$) can be embedded into H_{h+1} with *dilation* 1, *congestion* 2 and *load* 2.

Proof. Since DT_h can be embedded into H_{h+1}, we let two nodes on level 1 of DT_h embed respectively into the nodes on levels 0 and 2*h* of TM_h. Hence, the nodes between levels 0 and *h*-1 of TM_h are embedded into the left-subtree of the roots of DT_h. Likewise, the nodes between levels 2*h* and *h*+1 of TM_h are embedded into the right-subtree of the roots of DT_h (see Figure 10). Now, we discuss how to embed the nodes on level *h* of TM_h into H_{h+1}.

For two son nodes (on level *h*) of each node on level *h*-1 of TM_h, we embed one son node into its upper parent node (on level *h*-1) and embed the other son node into its lower parent node (on level *h*+1). The upper parent node has an edge to link to the lower parent node by Lemma 2. Hence, the dilation of two edges linking the node on level *h*-1 and its two son nodes on level *h* are respectively 0 and 1. Similarly, the dilation of two edges linking the node on level *h*+1 and its two son nodes on level *h* are respectively 0 and 1. The congestion of the edges between leaf x_i ($1 \leq i \leq 2^{h-1}$) and y_i of DT_h are 2 in H_{h+1}. The load of leaf nodes of DT_h are 2 in H_{h+1}. Therefore, TM_h is embedded into H_{h+1} with *dilation* 1, *congestion* 2 and *load* 2. □

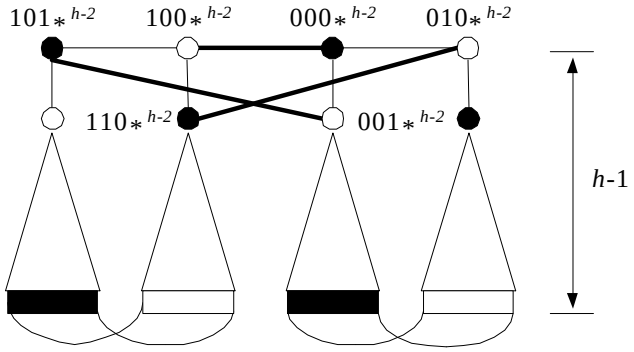


Figure 9. Construction of DT_h from two DT_{h-1}'s. The added edges are shown in solid lines. Nodes $000*^{h-2}$ and $100*^{h-2}$ are the double roots of DT_h.

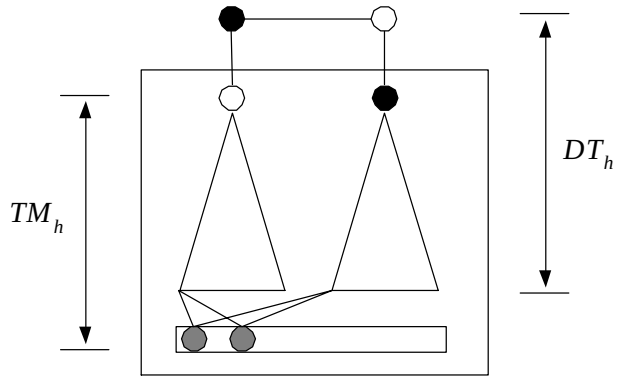


Figure 10. Embedding of TM_h into the DT_h contained in H_{h+1}.

A tree machine TM_h with $2^{h+1}+2^h-2$ nodes is embedded into H_{h+1} and this embedding of Theorem 2 is load-balanceable. Next, we show how to embed TM_h into H_n ($h \geq n$). First, we need three lemmas as follows.

Lemma 3. DT_{h-1} ($h \geq 1$) can be embedded into H_h and each leaf node of DT_{h-1} has an edge to link to a certain internal node of DT_{h-1} [16].

Next lemma shows how to embed a double-rooted complete binary tree of height h, DT_h, into H_h with load-balance.

Lemma 4. DT_h ($h \geq 1$) can be embedded into H_h with *dilation* 1, *load* 2 and *congestion* 2.

Proof. First, we embed DT_{h-1} into H_h, then we consider how to embed 2^h leaf nodes of DT_h into H_h as follows. Each leaf node of DT_h has to be embedded into different node in H_h to load-balance. There are half leaf nodes of DT_h to be embedded into their parent nodes (leaf nodes of DT_{h-1}) and remaining leaf nodes of DT_h are embedded to internal nodes of DT_{h-1} (see Figure 11). There is no dilation and congestion between half leaf nodes of DT_h and their parent nodes since these leaf nodes are embedded into their parent nodes. The dilation of the edges between remaining leaf nodes (embedded into internal nodes of DT_{h-1}) of DT_h and their parent nodes are 1 by Lemma 3. Hence, the *load* is 2 and the *dilation* is 1 for such embedding DT_h into H_h.

There are 2^{h-2} leaf nodes of DT_h being embedded into the internal nodes on level h-2 of DT_{h-1} for embedding 2^h leaf nodes of DT_h into H_h. These leaf nodes link to their parent nodes (on level h-1) to use the same edges as embedding DT_{h-1}, and the congestion of half edges between levels h-1 and h-2 of DT_{h-1} are 2 in H_h. Therefore, DT_h can be embedded into H_h with *dilation* 1, *load* 2 and *congestion* 2.

□

By Lemma 4, we show how to embed DT_h into H_h with load-balance. Next lemma describes the links between leaf nodes x_i and y_i ($1 \leq i \leq 2^{h-1}$) of DT_h for embedding DT_h into H_h.

Lemma 5. Embedding DT_h into H_h, leaf node x_i ($1 \leq i \leq 2^{h-1}$) is embedded into node X_i of H_h, and leaf node y_i is embedded into node Y_i of H_h. There is an edge to link node X_i and Y_i, and this edge is unused

for embedding the edges between levels h-1 and h-2 of DT_h into H_h.

Proof. We prove the lemma by induction on h.

Hypothesis: Embedding DT_{h-1} into H_{h-1}, leaf node x_i ($1 \le i \le 2^{h-2}$) is embedded into node X_i of H_{h-1}, and leaf node y_i is embedded into node Y_i of H_{h-1}. There is an edge to link node X_i and Y_i, and this edge is unused for embedding the edges between levels h-2 and h-3 of DT_{h-1} into H_{h-1}.

Basis step: When h=1 and 2, it is trivial. DT_3 is embedded into H_3 as shown in Figure 12. The figure describes the links between leaf nodes x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4.

Induction step: We partition H_h into two H_{h-1}'s by the most significant bit. By hypothesis, leaf node x_i ($1 \le i \le 2^{h-2}$) of DT_{h-1} is embedded into node X_i of H_{h-1}, and leaf node y_i of DT_{h-1} is embedded into node Y_i of H_{h-1}. There is an edge to link X_i and Y_i, and this edge is unused for embedding the edges between levels h-2 and h-3 of DT_{h-1} into H_{h-1}. Hence, we can merge two DT_{h-1}'s into DT_h as shown in Figure 13. The links between x_i ($1 \le i \le 2^{h-1}$) and y_i of DT_h are the same as the hypothesis since the three added edges in Figure 13 are extra edges that do not affect the links in the hypothesis. Therefore, the edge between levels h-2 and h-3 of both DT_{h-1}'s become to the edges between levels h-1 and h-2 of DT_h, and the lemma is proved. □

Theorem 3. A tree machine TM_h ($h \ge 1$) can be embedded into H_h with *dilation* 1, *congestion* 3 and *load* 3.

Proof. First, DT_h can be embedded into H_h with *dilation* 1, *congestion* 2 and *load* 2 by Lemma 4. We let two nodes on level 1 of DT_h embedded respectively into the nodes on level 0 and level 2h of TM_h. Then the nodes between levels 0 and h-1 of TM_h are embedded into the left-subtree of the roots of DT_h. Likewise, the nodes between levels 2h and h+1 of TM_h are embedded into the right-subtree of the roots of DT_h (see Figure 14). Next, we consider how to embed the nodes on level h of TM_h into H_h.

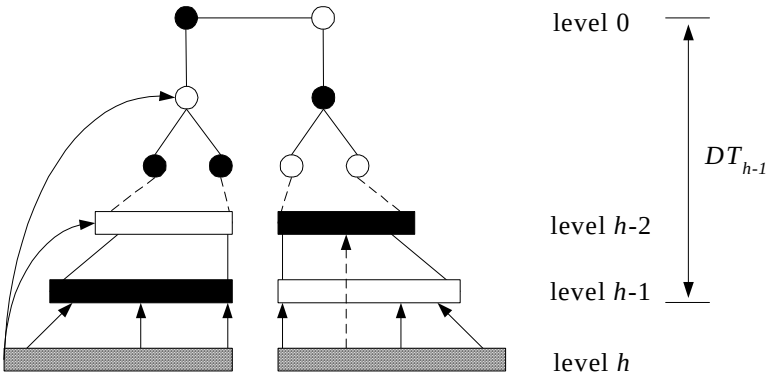


Figure 11. Embedding 2^h leaf nodes of DT_h into the DT_{h-1} contained in H_h. The leaf nodes of left subtree (right subtree) of DT_h are embedded into their parent nodes and white nodes (black nodes) between levels h-2 and 0, where $\rightarrow$ depicts embedding.

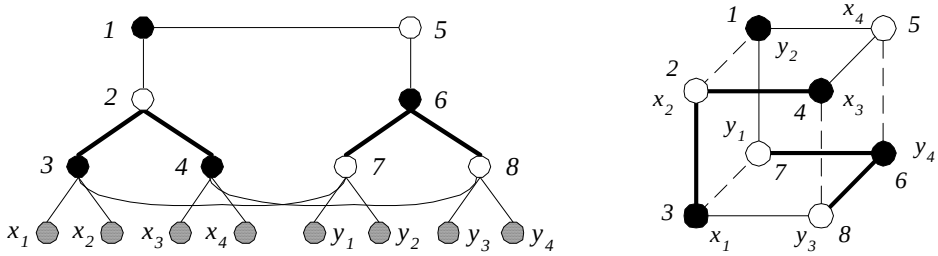


Figure 12. Nodes x_1, x_2, x_3, x_4, y_1, y_2, y_3 and y_4 are embedded into nodes *3*, *2*, *4*, *5*, *7*, *1*, *8* and *6*, respectively. Node x_i ($1 \le i \le 4$) has an edge (dashed line) to link to y_i, and this edge are unused for embedding the edges (bold lines) between levels 2 and 1 of DT_3 into H_3, where the congestion of bold lines are 2.

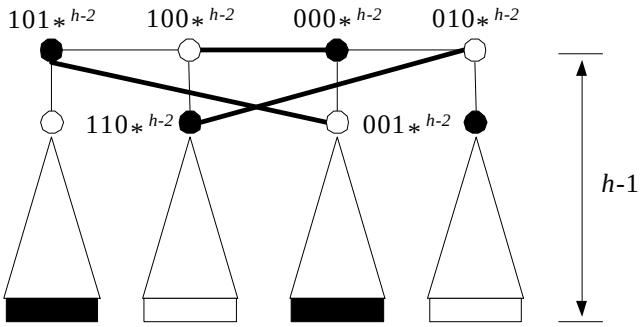


Figure 13. Construction of DT_h from two DT_{h-1}'s. The added edges are shown in solid lines. Nodes $000*^{h-2}$ and $100*^{h-2}$ are the double roots of DT_h.

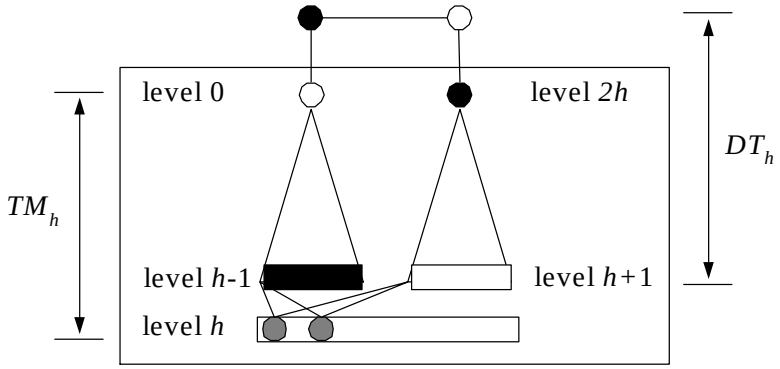


Figure 14. Embedding of TM_h into the DT_h contained in H_h.

For two son nodes (on level h) of each node on leve l h-1 of TM_h, we embed one son node into its upper parent node (on level h-1) and the other son node into its lower parent node (on level h+1). By Lemma 5, the upper parent node has an edge to link to the lower parent node and this edge is unused for embedding the edges between levels h-1 and h-2 of DT_h into H_h. The congestion of half edges between levels h-1 and h-2 are 2 and those of others are 1 by Lemma 4. Hence, the dilation of two edges linking the node on level h-1 and its two sons on level h are respectively 0 and 1. Similarly, the dilation of two edges linking the node on level h+1 and its two sons on level h are respectively 0 and 1. The congestion of edges linking between levels h-1 and h+1 of TM_h increase by 1 and become 3 since these edges may be used to embed DT_h into H_h. The *load* increases by 1, and it remains load-balance. Therefore, TM_h is embedded into H_h with *dilation* 1, *congestion* 3 and *load* 3. □

Theorem 4. A tree machine TM_h can be embedded into H_n ($h>n\geq 1$) with *dilation* 1, *congestion* 3 and *load* $2^{h-n+1}+2^{h-n}$.

Proof. First, we embed TM_h into TM_n; that is we embed the nodes between levels n and $2h$-n of TM_h into their ancestor nodes on level n of TM_h (see Figure 15). Hence, the load of each node on level n of TM_n is:

$$(2^{h+1}+2^h-2-2\times(2^n-1))/2^n=2^{h-n+1}+2^{h-n}-2$$

, and there is no dilation and congestion for this embedding. By Theorem 3, TM_n can be embedded into H_n with *dilation* 1, *congestion* 3 and *load* 3. Therefore, TM_h can be embedded into H_n with *load* $2^{h-n+1}+2^{h-n}-2+3-1=2^{h-n+1}+2^{h-n}$, the dilation and congestion are not altered. The theorem is proved. □

5 Conclusion

Öhring and Das [14] have described how to embed a tree machine of dimension h into an incomplete hypercube which comprises an (h+1)-dimensional hypercube and an h-dimensional hypercube with *load* 1, *dilation* 2, *congestion* 2 and *expansion* $(2^{h+1}+2^h)/(2^{h+1}+2^h-2)$. In this paper, we have shown

how to embed a tree machine into an incomplete hypercube with *load* 1, *dilation* 2, *congestion* 2 and *expansion* 1, and how to embed a large tree machine into a hypercube with optimal load.

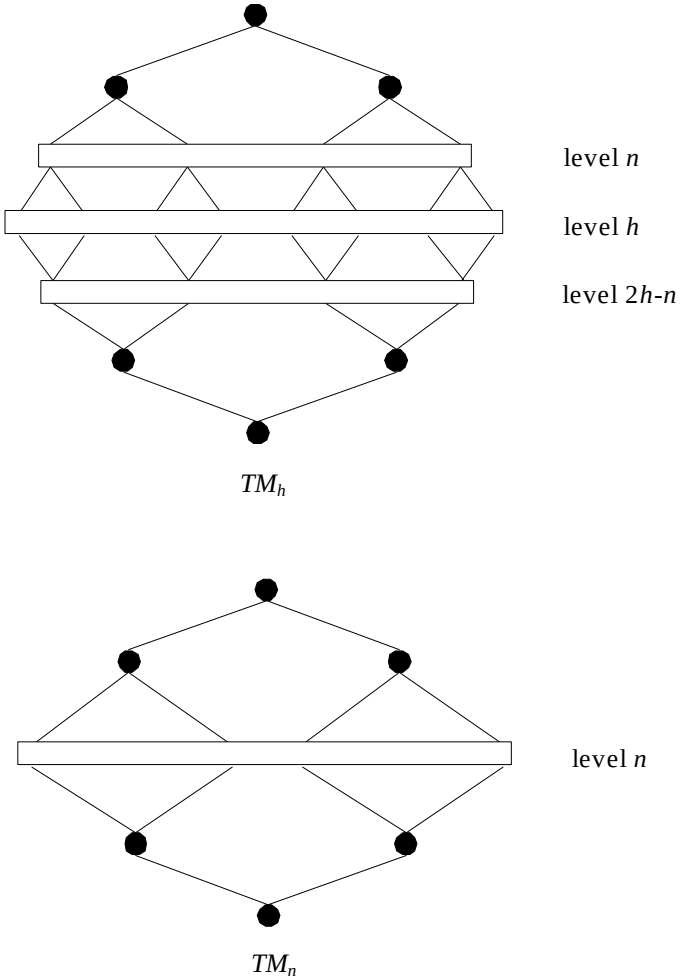


Figure 15. Embedding TM_h into TM_n.

Reference

[1] C. L. Seitz, The Cosmic Cube, *Commun, ACM* **28** (1985) 22-33.

[2] J. P. Hayes, T. N. Mudge, Q. F. Stout, S. Colley and J. Palmer, A Micoprocessor-based Hypercube Supercomputer, *IEEE Mico* **6** (1986) 6-17.

[3] Y. Saad and M. H. Schultz, Topological Properties of Hypercubes, *IEEE Trans. Computers* **C-37** (7) (1988) 867-872.

[4] D. A. Read and R. M. Fujimoto, *Multicomputer Network: Message-Based Parallel Processing*, Cambridge, MA: MIT Press (1987).

[5] T. Leighton, *Introduction To Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* (Morgan Kaufmann, Reading, MA, 1992).

[6] H. P. Katseff, Incomplete Hypercubes, *IEEE Trans. Computer* **37** (5) (1988) 604-608.

[7] H. C. Chen and N. F. Tzeng, Enhanced Incomplete Hypercubes, *Proc. Int. Conf. Parallel Processing* **1 (**1989) 270-277.

[8] N. F. Tzeng, Structural Properties of Incomplete Hypercubes, *Proc. 10th Int. Conf. Distrib. Comout. Syst.* (1990) 262-269.

[9] N. F. Tzeng, H. L. Cheng and P. J. Chuang, Embeddings in incomplete hypercubes, Proc. of Int. Conf. on Parallel Processing 3 (1990) 335-339.

[10] J. L. Bentley and H. T. Kung, A Tree Machine for Searching Problems, *Proc. IEEE 1979 Int. Conf. Parallel Processing* (1979) 257-266.

[11] S. A. Browning, The Tree Machine: a Highly Concurrent Computing Environment, *Tech. Rep. 1980: TR 3760*, Computer Science, California Institute of Technology (Jan. 1980).

[12] A. Gupta, A. Boals and N. Sherwani, On Optimal Embeddings into Incomplete Hypercubes, *Proc. the Fifth Int. Parallel*

Processing Symp. (1991) 416-423.

[13] K. Efe, Embedding Mesh of Trees in the Hypercube, *J. Parallel and Distrib. Comput.* **11** (1991) 222-230.

[14] S. Öhring and S. K. Das, Incomplete Hypercubes: Embeddings of Tree-Related Networks, *J. Parallel and Distrib. Comput.* **26** (1995) 36-47.

[15] A. L. Rosenberg, Issues in the Study of Graph Embeddings, In H. Noltemeir (ed.). *Graph-Theoretic Concept in Computer Science* (*Proc. Int. Workshop WG80*), Lecture Notes in Computer Science, Springer-Verlag, New York 100 (1981) 151-176.

[16] C. C. Chen and R. J. Chen, Optimal Embedding of Large Complete Binary Trees into Hypercubes, *J. of Information Science and Engineering* **12** (2) (1996) 307-314.